FREOPEN

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-22

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 8015 bytes

Attack Category	• Path spoofing or confusion problem
Vulnerability Category	• Indeterminate File/Path • TOCTOU - Time of Check, Time of Use
Software Context	• File Management • File Creation • File I/O
Location	• stdio.h
Description	The freopen() function first attempts to flush the stream and close any file descriptor associated with stream. Failure to flush or close the file successfully is ignored. The error and end-of-file indicators for the stream are cleared. The freopen() function opens the file whose pathname is the string pointed to by filename and associates the stream pointed to by stream with it. The mode argument is used just as in fopen(). freopen() is vulnerable to TOCTOU attacks. A call to freopen() should be flagged if the first argument (the directory or file name) is used earlier in a check-category call. On Windows platforms the APIs _freopen, _tfreopen, and _wfreopen are synonymous with freopen.
Method of Attack	The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

The freopen() call is a use-category call, which when preceded by a check-category call can be indicative of a TOCTOU vulnerability.

A TOCTOU attack in regards to freopen() can occur when

a. A check for the existence of the file or a non-fd reference (pathname) to the filename occurs

b. The actual call to freopen occurs.

Between a and b, an attacker could, for example, link the referenced file to a known file. The subsequent freopen() call would have an unintended effect or impact.

Exception Criteria

Solutions

Solution Applicability	Solution Description	Solution Efficacy
Generally applie tso any freopen() call.	Translate freopen() to fdopen if possible.	Effective.
Generally applie tso any freopen() call.	Translate freopen() to fdopen if possible.	Effective.
Generally applies to any freopen() call.	The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	Does not resolve the underlying vulnerability but limits the false sense of security given by the check.
Generally applies to any freopen() call.	Limit the interleaving of operations on files from	Does not eliminate the underlying vulnerability

		multiple processes.	but can help make it more difficult to exploit.
	Generally applies to any freopen() call.	Limit the spread of time (cycles) between the check and use of a resource.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.

Signature Details	FILE *freopen(const char *filename, const char *mode, FILE *stream); FILE *fopen(const char *path, const char *mode); FILE *fdopen(int fildes, const char *mode);
Examples of Incorrect Code	`int main () {` `struct stat stats;` `stat(path, &stats);` `...` `freopen ("myfile.txt","w",stdout);` `printf ("This sentence is redirected to a file.");` `fclose (stdout);` `return 0;` `}`
Examples of Corrected Code	/* freopen example: redirecting stdout */ /* no check performed */ `int main ()` `{` `freopen ("myfile.txt","w",stdout);` `printf ("This sentence is redirected to a file.");` `fclose (stdout);` `return 0;` `}`
	`/* fdopen version */` `main() {` `char fn[]="fdopen.file";` `FILE *stream;` `int fd;` `if ((fd = creat(fn, S_IWUSR)) < 0) perror("creat() error");` `else {` `if ((stream = fdopen(fd, "w")) == NULL) {` `perror("fdopen() error");`

	`` close(fd); } else { fputs("This is a test", stream); fclose(stream); } unlink(fn); } } ```
Source References	• Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, ch. 9 • freopen() man page • Microsoft Developer Network Library
Recommended Resource	
Discriminant Set	**Operating System**: • UNIX **Languages**: • C • C++

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com